Project plan

In this project we will create a dungeon crawler game in C++. This project plan will serve as a guideline for the group members and will contain our initial plans for the project. As the project plan is done early on in the project, it might be subject to some changes as the project proceeds.

Communication

The communication between group members will be mainly held through discord, where a server dedicated to the project has been created. Meetings will be held weekly on Mondays at 2 pm. Additional meetings will be held at other times when deemed necessary. At least during the first weeks we assume that extra meetings will be required to get the project going. To further improve communication between group members, it was agreed that everyone could join a voice channel if they are working on the project so that another member can join them if they are also working on the project.

Game description

The dungeon crawler will be realtime so that the player can continuously move around and attack monsters. The game will be roguelike so that when you die you lose all or atleast most of your gained items and levels. The game starts from a main room connected to 3 different doors (less or more, depending on how much time we have on creating different levels). Each door leads to a new dungeon with difficulty increasing with each door. On the levels you are able to boost your character's strength with items and abilities to help you on your way to beat all levels. Going with the roguelike theme, you lose most of your boosts upon death, but some buffs might be permanent, but also optional. So the more you play, the stronger your character becomes.
Each level has a win condition, which might be just reaching the end or beating a boss. Getting this win-condition returns you to the starting room, allowing you to go to the next level. Upon completing all the levels you reach the end of the game.

In the project plan folder we have provided our quickly drafted and crude class diagram, where we have thought about what kind of main classes are going to be required but we haven’t for the most part specified methods, and only some variables. This diagram is

subject to change. A lot during development, when we have time to better plan the structure of the program.

Features

Required features

- Simple 2D graphics
    - We will use SFML to implement our 2D graphics. We have already experimented with the library and are able to load sprites from image files into our game window.
    - We must at least draw the player, walls, monsters, chests, shops, projectiles to the game window.
    - We want to implement an animated walk-cycle for our player character.
    - We will try to have some sort of attack and/or sprint animation for our player
- Moving through corridors and rooms.
    - We have decided to make our game camera center/focus top down on one room at a time. When moving to another room the camera will then move to the new room. The camera will not follow the player.
- Combat between the player and different types of monsters
    - We want to have both melee and ranged combat in our game.
    - We want to make the player be able to switch his weapon to weapons that he finds or buys.
    - Some sort of collision system is needed for the projectiles.
    - We may want to add a spring/dash ability, depending on time and how it fits into the game.
- Collectibles which can be used later on (weapons, potions, etc.)
    - Most collectibles will probably be acquired from the shop
    - We will at least implement a couple of different weapons and most likely a health potion.
- Some sort of progression (winning & losing conditions)
    - You die when you have no hit points left
    - The player will need to complete all the levels to win. Each level has some sort of winning condition. The condition can be defeating the boss or just reaching the end of the level.

Advanced features

We want to try to implement at least in some way all of the following features. Depending on time and problems that will emerge we might have to scale back the project. We have the most important features that we want to implement first in bold.

- Randomly generated dungeons
    - We will define a set of room types, e.g., shop-room and chest-room. We can then place some of these rooms at random positions on each floor.
- **Randomly generated monsters and items**
    - We can make a couple of different monster types. When walking into a new room we will spawn a random constrained amount of different monsters, also depending on the difficulty of the floor. Some rooms can also be safe -> No monsters will spawn.
    - Monsters will not respawn
- **Character leveling (skills & abilities)**
    - We are going to implement some sort of leveling up system. It will probably be based on experience gained from killing monsters.
    - Some of the buffs gained from playing will be lost when dying and some may be held when respawning.
- Quests or other events
    - Simple achievements could be added by keeping track of attacks, movement, enemies killed, etc. Example: die 10 times, kill 20 enemies without taking any dmg.
- Other kinds of non-player characters (**shopkeepers**, allied combat, etc.)
    - We are planning to have the shopkeeper be located in the starting room. The player will then return to the start to make purchases with the money collected.
    - We might also have some smaller shops located in the levels. They could sell some frequently needed items like healthpotions.
- Additional UI elements (dungeon map, interactive inventory, etc.)
    - We will investigate how some of these could be implemented if we decide we have enough time for it.
- Sound effects
    - Simple sound effects, death sounds, win condition jingle and etc, should easily be added through the SFML library functions.

Libraries

To implement our game we will use the SFML library recommended in the project instructions. We believe that SFML provides all necessary functions for our game that the standard c++ libraries do not provide, such as graphics, audio, event handling, and parsing of user input. If we decide to allow saving in our game we want to use some sort of JSON or XML library for easier retrieval of the game-state, but saving is not a priority-feature for us.

Responsibilities of group members

Below we list what each group member will start working on in the initial stage. The features not listed below will be done by a suitable member depending on the success of the earlier tasks.

Oskar: Game design and visuals
Henrik: Software architecture
Jonas: The character classes.
Mikael: Game cycle and movement

We will on a weekly basis assign tasks to different team members so that everyone has some responsibility to work each week.

Schedule

- Week 1
    - Project plan submitted on Friday 5.11.
    - When the project plan is submitted everyone should have a basic understanding of what should be done and how the used libraries and tools work.
    - Everyone should be familiar with the project workflow and git.
- Week 2
    - A very basic working version of the game ready. Contains some very basic room where some very basic player can move. We stress that this version will be very basic and will serve as a platform for future improvements.
- Week 3
    - Implementing items, enemies and walls.
- Week 4

    - Implementing features related to items. Creating floors from the rooms.

- Week 5
    - Implementing the more advanced features which might require more time than we have. Hopefully we will have a version of the game that would be good enough as a final version.
- Week 6
    - Making sure that everything works. Start to write documentation and finish testing.